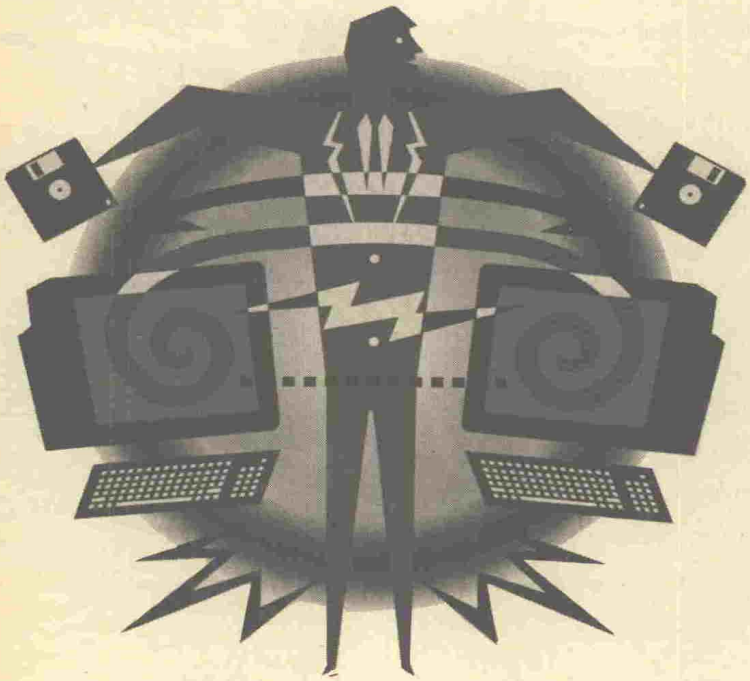
เรื่องน่าอ่าน
สมชาย นำประเสริฐชัย

สารพัน UNIX ตอน : การพัฒนาโปรแกรม



การพัฒนาโปรแกรมบนระบบยูนิกซ์เพื่อสามารถเลือกใช้ได้ตรงกับงาน

ก่อนอื่นขอกล่าวคำว่าสวัสดีครับกับท่านผู้อ่านที่ติดตามและให้ความสนับสนุนคอลัมน์สารพัน...UNIX มาตลอด หนึ่งปีกว่าแล้วนะครับที่สารพัน...UNIX ของเราได้นำเสนอเรื่องราวต่าง ๆ เกี่ยวกับยูนิกซ์ที่น่าสนใจมากบ้างน้อยบ้างหนักบ้างเบาบ้างคละเคล้ากันไปสำหรับสารพัน...UNIX ฉบับนี้เป็นเรื่องของการพัฒนาโปรแกรมบนระบบยูนิกซ์ที่หลายคนมีมุมมองที่แตกต่างกันไปยากบ้าง ง่ายบ้าง ตามแต่ความถนัดของแต่ละคน

ยูนิกซ์เป็นโอเอสที่มียูทิลิตี้ต่าง ๆ จำนวนมากที่สนับสนุนการใช้งานของผู้ใช้งานในระบบแต่อย่างไรก็ตามยูทิลิตี้ที่อยู่นี้ก็ไม่สามารถที่จะสนับสนุนการทำงานทุกประเภทของผู้ใช้ทุก ๆ ลักษณะงานได้โดยตรง ยูทิลิตี้ของระบบไม่ได้ออกแบบมาเพื่องานในประเภทใดประเภทหนึ่งโดยเฉพาะ ดังนั้นการใช้งานยูทิลิตี้บางอย่างนั้นจะต้องนำยูทิลิตี้มาประยุกต์ใช้งานในรูปแบบที่เหมาะสม อาจเป็นการนำยูทิลิตี้หลาย ๆ ตัวมาเรียงต่อเข้าด้วยกันตามลำดับการทำงานก่อนหลังเป็นสคริปต์ หรืออาจใช้โปรแกรมภาษาคอมพิวเตอร์สร้างยูทิลิตี้หรือโปรแกรมประยุกต์ขึ้นใหม่เพื่อให้เกิดความเหมาะสมกับงานนั้น ๆ มากยิ่งขึ้น

รูปแบบและขั้นตอน

การพัฒนาโปรแกรมมิใช่เรื่องง่าย แต่ก็ไม่ยากจนเกินไป ปัญหาในการพัฒนาโปรแกรมก็คือผู้พัฒนาโปรแกรมหรือผู้ใช้งานทั่วไปเมื่อต้องการพัฒนาโปรแกรมหรือประสบปัญหาเมื่อไหร่ก็พยายามตั้งหน้าตั้งตาเขียนโปรแกรมเพื่อให้บรรลุจุดประสงค์เหล่านั้นทันทีโดยไม่ได้คำนึงถึงขั้นตอนและหลักการของการพัฒนาโปรแกรม

หลักการของการพัฒนาโปรแกรมจะมีลำดับขั้นตอนต่าง ๆ เพื่อช่วยให้นักพัฒนาโปรแกรมสามารถสร้างและพัฒนาโปรแกรมทำได้อย่างมีระเบียบแบบแผนเพื่อให้งานที่ปรากฏออกมามีประสิทธิภาพยิ่งขึ้นโดยแบ่งลำดับขั้นตอนเป็นดังนี้

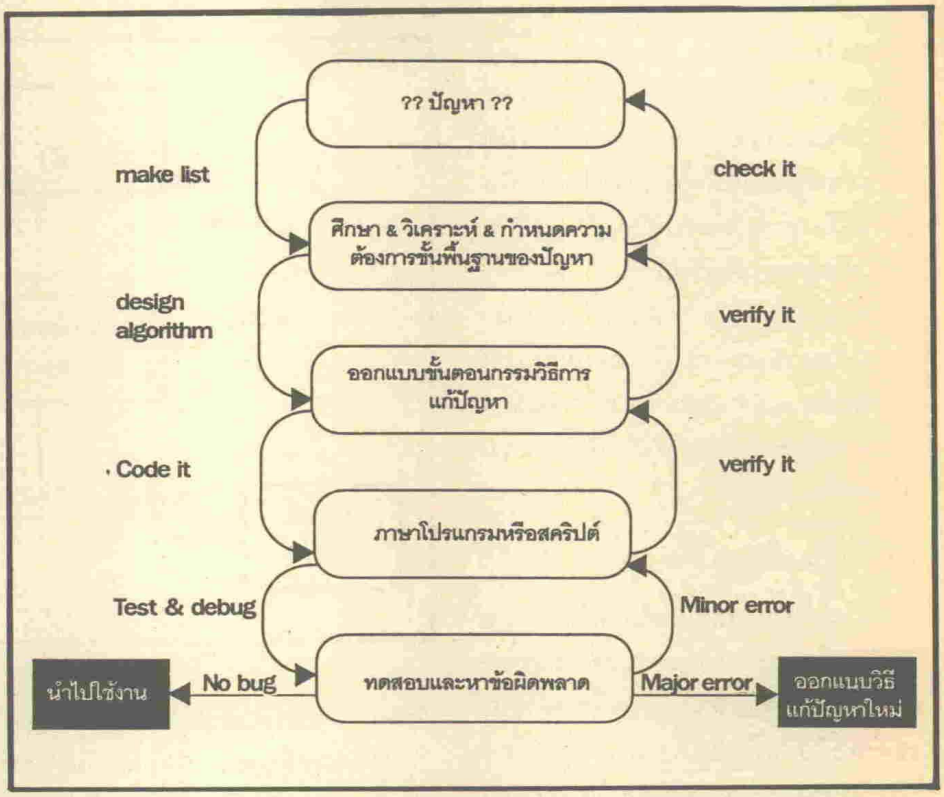
1. ศึกษาและวิเคราะห์ปัญหารวมถึงความต้องการขั้นพื้นฐานของปัญหานั้น ๆ (Determine & Analyze Problems)

2. ออกแบบขั้นตอนกรรมวิธีในการแก้ปัญหา (Design an Algorithm)

3. เปลี่ยนขั้นตอนที่ 2 ให้อยู่ในรูปแบบภาษาหรือสคริปต์ที่ระบบสามารถทำงานตามความต้องการได้ (Coding)

4. ทดสอบและตรวจหาข้อผิดพลาดของโปรแกรม (test & debug)

5. ทำเอกสารประกอบโปรแกรมทั้งการใช้งานและเทคนิค (User & Technical Manual) ดังรูปที่ 1



รูปที่ 1 ขั้นตอนการพัฒนาโปรแกรม

ทำไมยังต้องใช้ภาษาคอมพิวเตอร์

ระบบยูนิกซ์มีเชลล์หลายชนิดรองรับคำสั่งการทำงานของผู้ใช้คือมีทั้งบอร์นเชลล์ ซีเชลล์ คอร์นเชลล์ โดยเฉพาะอย่างยิ่งคอร์นเชลล์เป็นเชลล์ที่มีความสามารถสูงสามารถเขียนเป็นโปรแกรมที่เรียกว่าเชลล์สคริปต์ได้ดังตอนที่ผ่านมา แต่การทำงานของเชลล์นั้นมีลักษณะเป็นการตีความคำสั่ง (Command interpreter) ซึ่งจะต้องมีการตีความใหม่ทุก ๆ ครั้งที่มีการเรียกใช้งาน ซึ่งระบบจะต้องสูญเสียเวลาในการตีความทำให้การใช้งานไม่สะดวกเท่าไรนัก ระบบยูนิกซ์จึงให้มีการพัฒนาโปรแกรมอีกรูปแบบหนึ่งโดยใช้โปรแกรมภาษาคอมพิวเตอร์ทั่วไป ระบบยูนิกซ์ได้เตรียมเครื่องมือต่าง ๆ ไว้คอยอำนวยความสะดวกแก่ผู้ใช้ในการพัฒนางานหลายอย่างเช่น พรีโปรเซสเซอร์ (preprocessor) แอสเซมเบลอร์ (Assembler) คอมพายเลอร์ (Compiler) และลิงค์เกอร์ (linker) แต่ทั้งหมดนี้ไม่ได้มีอยู่ในยูนิกซ์ทุกระบบ แต่ยูนิกซ์เกือบทุกระบบได้เตรียมเครื่องมือและยูทิลิตี้ต่าง ๆ สำหรับการพัฒนาโปรแกรมด้วยภาษาซีไว้แล้ว

ยูนิกซ์กับภาษาซี

ภาษาซีเป็นภาษาระดับกลางที่มีความยืดหยุ่นในการใช้งานและโยกย้ายชิ้นงานไปทำงานในเครื่องระดับต่าง ๆ ได้ ภาษาซีมีเครื่องมือและโปรแกรมอำนวยความสะดวกต่าง ๆ มากมาย ระบบยูนิกซ์และภาษาซีมีความสัมพันธ์กันอย่างใกล้ชิดและตัวระบบยูนิกซ์เองถูกสร้างด้วยภาษาซี จึงไม่ใช่เรื่องแปลกที่ระบบยูนิกซ์ให้การสนับสนุนการเขียนโปรแกรมภาษาซีเป็นพิเศษเมื่อเทียบกับภาษาโปรแกรมอื่น ๆ

ในระบบยูนิกซ์ยังสามารถกำหนดให้โปรแกรมภาษาซีทำงานร่วมกับคำสั่งของเชลล์ของระบบหรือเชลล์สคริปต์ได้โดยกำหนดคำว่า system นำหน้าเชลล์สคริปต์ที่ต้องการให้ทำงานร่วมด้วย

```
main( )
{
system ("shell script");
}
```

เครื่องมือในการพัฒนา

1) เอดิเตอร์วีไอ

การสร้างและแก้ไขเป็นฟังก์ชันหลักที่ใช้ในระบบคอมพิวเตอร์ การแก้ไขข้อมูลเป็นสิ่งที่หลีกเลี่ยงไม่ได้ถึงแม้ว่าจะมีการวางแผนมาดีขนาดไหนก็ตาม ข้อผิดพลาดย่อมเกิดขึ้นได้ ในระบบยูนิกซ์ให้ความสำคัญกับการสร้างไฟล์ขึ้นใหม่และการแก้ไขเปลี่ยนแปลงไฟล์เดิมที่มีอยู่แล้ว โดยในระบบยูนิกซ์มีเอดิเตอร์ถึง 2 คือเอดิเตอร์อีดี (ed editor) จะเป็นเอดิเตอร์ในลักษณะของ

เอดิเตอร์แบบบรรทัด (line editor) และเอดิเตอร์วีไอ (VI editor) เป็นเอดิเตอร์แบบเต็มหน้าจอ ซึ่งเอดิเตอร์วีไอนี้มีบทบาทมากและมีการนำไปใช้งานกันอย่างกว้างขวางสำหรับหรือแก้ไขเปลี่ยนแปลงข้อมูลภายในไฟล์

เอดิเตอร์วีไอ พัฒนาโดยมหาวิทยาลัยคาลิฟอร์เนีย แห่งเบริกเลย์ (University of California at Berkeley) โดยตัวเอดิเตอร์นี้ถูกนำไปใช้ทั้งในระบบยูนิกซ์และซีนิกซ์

เริ่มต้นกับเอดิเตอร์วีไอ

การใช้งานเอดิเตอร์หรือโปรแกรมทั่วไปเกือบทุกประเภทจะต้องมีการกำหนดค่าของสภาวะแวดล้อมต่าง ๆ เอดิเตอร์วีไอก็เช่นเดียวกัน จะต้องกำหนดชนิดของเทอร์มินัล โดยต้องทำการกำหนดค่าทางเซลล์

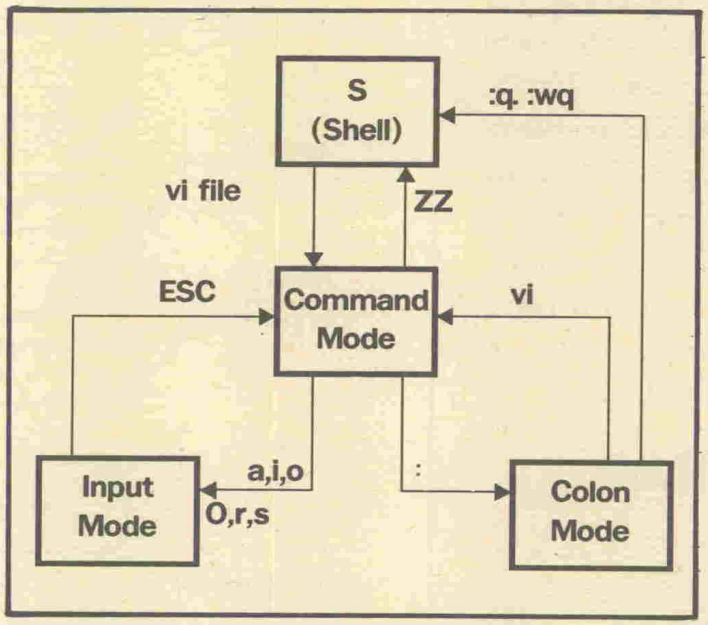
```
$echo $TERM
$TERM = ? (ประเภทของ terminal)
$export TERM
```

โดยส่วนใหญ่แล้วการติดตั้งชนิดของเทอร์มินัลนั้น จะทำการติดตั้งเป็นค่าปกติไว้ในไฟล์ profile ได้ หากการติดตั้งไม่ตรงกับชนิดของเทอร์มินัลจะต้องทำการกำหนดค่าใหม่ เพราะว่าเอดิเตอร์วีไอ จะไม่สามารถทำงานเป็นเอดิเตอร์แบบเต็มจอภาพได้

โหมดการทำงาน

ในเอดิเตอร์วีไอนั้น มีการทำงานอยู่ 3 โหมด ซึ่งตัวคีย์จะมีความหมายแตกต่างกันในแต่ละโหมดขึ้นอยู่กับว่าขณะนั้นทำงานอยู่ในโหมดใดดังรูปที่ 2

โหมดคำสั่ง (command mode) จะเป็นโหมดที่ทำงานเกี่ยวกับการเคลื่อนย้ายเคอร์เซอร์ลบตัวอักษร



รูปที่ 2
โหมดการทำงานของเอดิเตอร์วีไอ

โหมดอินพุต (input or text mode) จะเป็นโหมดในการใส่ข้อมูลลงใน text file ซึ่งจะสิ้นสุดเมื่อการทำงานในโหมดนี้มีการกด ESC

โหมดโคลอน (colon mode) เป็นการใช้คำสั่งจากเอดิเตอร์ตัวเก่า

การเข้าใช้งานเอดิเตอร์

การเข้าสู่เอดิเตอร์วีไอเป็นเรื่องที่ง่ายมากโดยการพิมพ์คำสั่ง vi หรือ vi แล้วตามด้วยชื่อไฟล์ที่ต้องการสร้างขึ้นใหม่หรือไฟล์ที่มีอยู่เดิมต้องการแก้ไข

```
$vi filename
```

โดย filename เป็นชื่อไฟล์ที่ต้องการแก้ไข หรือเป็นไฟล์ใหม่ที่ต้องการสร้างจากเอดิเตอร์วีไอ แต่สำหรับถ้าเป็นไฟล์ที่มีอยู่แล้ว การใช้เอดิเตอร์วีไอเพื่อกระทำแก้ไขเปลี่ยนแปลงนั้นจะทำได้ ก็ต่อเมื่อผู้ใช้รายนั้นมีสิทธิ์ในการดำเนินการแก้ไขกับไฟล์นั้นด้วย

ลักษณะของจอภาพภายในเอดิเตอร์วีไอ จะมีเครื่องหมาย tildes (~) เป็นตัวบอกบรรทัดและในส่วนล่างของจอภาพจะมีข้อความแสดงข้อมูลบางอย่างเกี่ยวกับไฟล์เช่นดังรูปที่ 3 ข้อความจะขึ้น "Filename" (New File) ในกรณีที่เป็นไฟล์สร้างขึ้นใหม่และดังรูปที่ 4 ข้อความจะขึ้น ".profile" 9 lines, 254 characters ในกรณีที่เป็นไฟล์เก่าที่มีอยู่แล้วแต่นำมาทำการแก้ไขเปลี่ยนแปลง

การใส่และการลบข้อความ

การเข้าสู่เอดิเตอร์วีไอ ครั้งแรกเป็นการเข้าสู่โหมดคำสั่ง ถ้าผู้ใช้ต้องการเขียนข้อความจากเอดิเตอร์จะต้องใช้คำสั่ง i (Insert) หรือ a (append) ก่อนจึงจะสามารถเขียนข้อความเพิ่มเติมลงเอดิเตอร์ได้ ลักษณะความแตกต่างของคีย์ทั้งสองนี้คือ i เป็นการเติมข้อมูลในตำแหน่งข้างหน้าของเครื่องหมายเคอร์เซอร์ ส่วน a นั้นเป็นการเพิ่มเติมข้อความใหม่เข้าไปที่หลังตำแหน่งของ

ต้นฉบับไม่มีหน้า 179-182

No This Page 179-182 In Original

เครื่องหมายเคอร์เซอร์นอกจากนี้ยังมีคำสั่งในการเติมในส่วนต้นและท้ายของแต่ละบรรทัดด้วยคือ I และ A

การลบข้อความในเอดิเตอร์วีไอนั้น จะต้องทำในโหมดคำสั่งเหมือนกับการเพิ่มเติมข้อความ ผู้ใช้งานสามารถเลือกรูปแบบในการลบข้อความได้โดย คีย์ x นั้นเป็นการลบข้อความทีละ 1 ตัวอักษรที่อยู่ในตำแหน่งหน้าเครื่องหมายเคอร์เซอร์ dd เป็นการลบข้อความทั้งบรรทัดที่เครื่องหมายเคอร์เซอร์อยู่ นอกจากนี้ยังมีการลบเป็นคำและลบข้อความทั้งหมดที่อยู่หลังเครื่องหมายเคอร์เซอร์ในบรรทัดนั้น

การควบคุมเคอร์เซอร์

ขณะที่อยู่ในโหมดของการเพิ่ม ผู้ใช้ไม่สามารถควบคุมการเคลื่อนของ cursor ในแนวขึ้นลงซ้ายหรือขวาได้ การที่จะควบคุมเปลี่ยนตำแหน่งของ cursor ได้นั้นจะต้องอยู่ในโหมดคำสั่งก่อนแล้วจึงสามารถควบคุมทิศทางของ cursor ได้โดยใช้คีย์พื้นฐาน 4 คือ

h เคลื่อนย้ายเครื่องหมายเคอร์เซอร์ไปทางซ้าย

l เคลื่อนย้ายเครื่องหมายเคอร์เซอร์ไปทางขวา

k เคลื่อนย้ายเครื่องหมายเคอร์เซอร์ขึ้นข้างบน 1 บรรทัด

j เคลื่อนย้ายเครื่องหมายเคอร์เซอร์ลงข้างล่าง 1 บรรทัด

การออกไปใช้งาน shell ชั่วคราว

ขณะที่ทำงานอยู่ในเอดิเตอร์วีไอนั้น ผู้ใช้สามารถเรียกคำสั่งจากเชลล์ขึ้นมาทำงานได้ โดยเข้าสู่โหมดโคลอนก่อน แล้วให้นำหน้าคำสั่งเชลล์ที่ต้องการใช้งานด้วยเครื่องหมาย ! ส่วนการกลับสู่การใช้งานเอดิเตอร์ใหม่อีกครั้งนั้นเพียงแต่ผู้ใช้กดคีย์อะไรก็ตามบนแป้นพิมพ์

```
:!pwd
```

การกำหนดจำนวนครั้งในการทำงาน

เอดิเตอร์วีไอ สามารถให้ผู้ใช้กำหนดจำนวนครั้งในการทำงานแต่ละคำสั่งได้โดยพิมพ์จำนวนครั้งไว้ที่หน้าคำสั่งที่ต้องการเช่น 7x เป็นการลบ 7 ตัวอักษรที่ตำแหน่งหน้าเคอร์เซอร์อยู่

ยกเลิกคำสั่ง

ในเอดิเตอร์วีไอ ถ้าหากผู้ใช้สั่งงานผิดพลาดไปแล้วต้องการยกเลิกคำสั่งนั้นสามารถทำได้ 2 รูปแบบคือ u (undo) เป็นการยกเลิกคำสั่งที่เพิ่งดำเนินการไปก่อนหน้านี้และ U (undo) ยกเลิกการดำเนินการทั้งหมดที่กระทำในบรรทัดนั้น โดยมีเงื่อนไขว่าจะต้องไม่มีการ

```
~
~
~
~
~
~
~
"filename" [New File]
```

รูปที่ 3
แสดงจอภาพในกรณีที่เป็นไฟล์ที่สร้างขึ้นใหม่

```
PATH = /usr/bin:/etc:/usr/sbin:/usr/ucb:$HOME/bin:/usr/bin/X11
:/sbin:

export PATH

if[ -s"$MALL"] # This is at Shell startup. In normal

then echo "$MAILMSG" # operation, the Shell checks
fi                    # periodically
~
~
~
".profile" 9 lines, 254 characters
```

รูปที่ 4 แสดงจอภาพกรณีที่นำไฟล์เก่ามาทำการแก้ไขเปลี่ยนแปลง

ย้ายเครื่องหมายเคอร์เซอร์ไปทำงานในบรรทัดอื่นด้วย

ค้นหาและแทนที่

เอดิเตอร์วีไอเหมือนกับเอดิเตอร์ทั่วไปคือ มีความสามารถค้นหาและย้ายตำแหน่งไปยังตำแหน่งนั้นหรือแทนค่าด้วยค่าอื่นก็ได้ โดยมีสัญญาที่ใช้ 2 ค่าคือ ? และ / แล้วตามด้วยข้อความที่ต้องการค้นหาส่วนการค้นหาและให้แทนค่าด้วยค่าอื่นนั้น จะต้องมีการระบุขอบเขตของข้อมูลที่ต้องการเปลี่ยนแปลงและรูปแบบการเปลี่ยนว่าจะเปลี่ยนเป็นอะไรโดยมีรูปแบบเป็น

: ~ ขอบเขต ~ S/ข้อความเก่า/ข้อความใหม่ เช่น

: 4, 10 s/use /student

การออกจากเอดิเตอร์

การออกจากโหมดของเอดิเตอร์วีไอ สามารถกระทำได้หลายลักษณะขึ้นอยู่กับความต้องการของผู้ใช้ว่าต้องการทำอะไรกับไฟล์ที่อยู่บนเอดิเตอร์วีไอขณะนั้น โดยอาจจะให้มีการเก็บไฟล์หลังจากแก้ไขหรือเก็บไฟล์แล้วออกจากโหมดของเอดิเตอร์วีไอ หรือออกจากโหมดของเอดิเตอร์วีไอ โดยไม่ทำการแก้ไขใด ๆ เลย

:w เขียนไฟล์

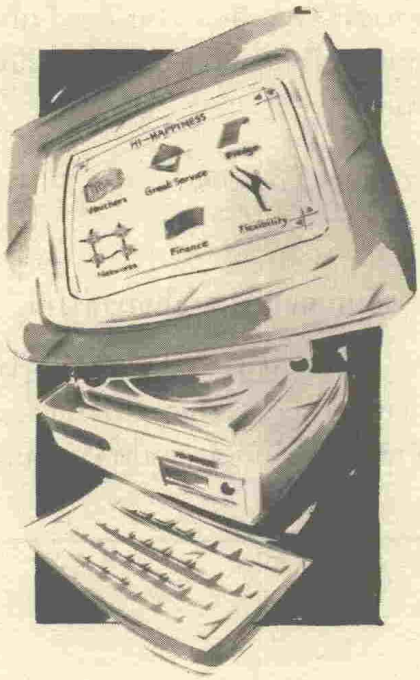
:q ออกจากวีไอโดยไม่ได้แก้ไขข้อมูลเดิมเลย

: q! ออกจากวีไอโดยได้ทำการแก้ไขไปบ้างแล้ว

:wq เขียนไฟล์และออกจากวีไอ

ข้อเด่นของวีไอเอดิเตอร์

เอดิเตอร์วีไอ เป็นเอดิเตอร์แบบเต็มจอภาพ ที่มีอยู่ในระบบยูนิกซ์ทุกระบบแล้ว มีการใช้งานที่คล่องตัว เนื่องจากมีคีย์หรือปุ่มควบคุมหลัก ๆ จำนวนน้อยและมีความสามารถเพียงพอในการใช้งาน



2) โปรแกรมคอมพายเลอร์ภาษาซี (CC)

โปรแกรม cc (C Compiler) เป็นโปรแกรมแปลภาษาซีซึ่งนับว่าเป็นส่วนสำคัญในการพัฒนาโปรแกรม โปรแกรมภาษาซีพื้นฐานจะเป็นการแปลรูปแบบของไฟล์ต้นฉบับที่มีรูปแบบตามข้อกำหนดของภาษาให้เป็นภาษาที่ระบบสามารถนำไปทำงานได้ การใช้งานโปรแกรม CC นี้สามารถเรียกใช้งานได้เหมือนกับคำสั่งในระบบคำสั่งหนึ่งเลยทีเดียว

รูปแบบการใช้งานของโปรแกรม CC

ส่วนของออปชันเป็นส่วนกำหนดรูปแบบพิเศษในการทำงานของโปรแกรม CC ตามความต้องการของผู้ใช้งาน

−0 <output file> กำหนดชื่อไฟล์ผลลัพธ์ที่ได้จากการแปล เพราะโดยปกติแล้วหากไม่มีการเติมออปชัน −0 นี้ระบบจะกำหนดไฟล์ผลลัพธ์เป็นไฟล์ชื่อ a.out ทุกครั้ง

−D <name> [= <value>] เป็นการกำหนดค่ามาโครให้กับ <name> โดยมีผลเหมือนกับคำสั่ง # defind ในส่วนของโปรแกรม

−l <link option> เป็นการนำไลท์บรารีมาเชื่อมกับส่วนของโปรแกรม

−O กำหนดให้โปรแกรมทำการปรับผลลัพธ์ที่ได้ให้เหมาะสมที่สุด (optimization)

−E กำหนดให้ทำการแปลเฉพาะส่วนของพรีโพรเซสเซอร์เท่านั้น

−S กำหนดให้แปลอย่างเดียวให้ได้ไฟล์ผลลัพธ์เป็นภาษาแอสเซมบลี

−C กำหนดให้ทำการแปลอย่างเดียวไม่ต้องลิงค์

3) Make

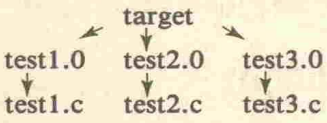
การพัฒนาโปรแกรมที่มีขนาดใหญ่มักจะมีการแบ่งงานออกเป็นส่วน ๆ ในรูปแบบของโปรแกรมย่อยก่อนแล้วจึงค่อยนำโปรแกรมย่อยเหล่านั้นมาทำการเชื่อมเข้าด้วยกันใหม่ โปรแกรมย่อยแต่ละส่วนจะมีลักษณะเป็นกลุ่มของข้อมูลในรูปแบบของภาษาคอมพิวเตอร์ บางส่วนอาจจะมีการใช้ข้อมูลร่วมกัน หากโปรแกรมได้รับการเปลี่ยนแปลงแล้วยิ่งเป็นชิ้นงานขนาดใหญ่เป็นการยากที่จะติดตามไฟล์ทุก ๆ ไฟล์ที่มีการแก้ไข

Make เป็นยูทิลิตี้ที่ช่วยในการจัดการชิ้นงานที่เกิดจากการรวมหลาย ๆ โปรแกรมเข้าด้วยกันโดยมีรูปแบบการ

เก็บข้อมูลที่เป็นอิสระต่อกัน ถ้าไม่มี make แล้วหากมีการเปลี่ยนแปลงส่วนใดส่วนหนึ่งของโปรแกรม ทุก ๆ ส่วนของไฟล์จะต้องทำการคอมพายล์ และลิงค์ใหม่

การทำงานของโปรแกรม make นั้นต้องอาศัยไฟล์ Makefile ซึ่งอาจมีชื่อเป็น makefile หรือ Makefile ซึ่งเป็นไฟล์ที่เก็บข้ออธิบายความสัมพันธ์ระหว่างโปรแกรมต้นฉบับต่าง ๆ

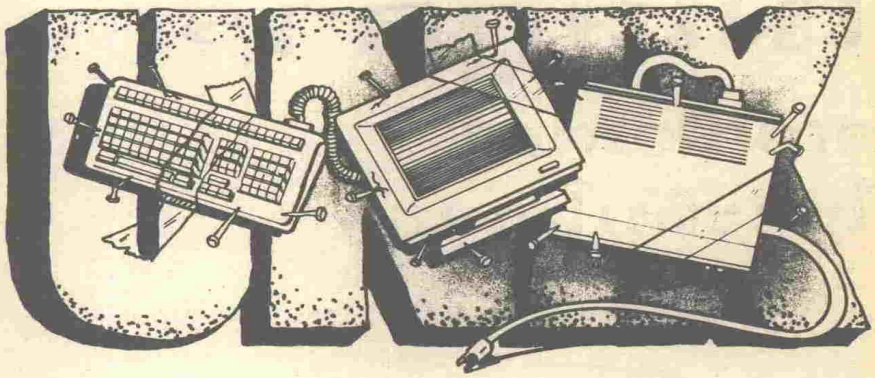
คำสั่ง make จะทำการอ่านไฟล์ makefile และดำเนินการตามเพื่อให้ได้ไฟล์เป้าหมายตัวอย่างเช่นไฟล์ปลายทางประกอบด้วยไฟล์ออปเจ็กต์ 3 ไฟล์ คือ test1.0 test2.0 และ test3.0 ซึ่งไฟล์ออปเจ็กต์แต่ละไฟล์ได้มาจากการคอมพายล์ไฟล์โปรแกรมภาษาซี



ผู้ใช้สามารถเรียก make เพื่อทำงานได้เลยโดยมี makefile เป็นสคริปต์ในการทำงาน ดังตัวอย่าง

รูปแบบของ Makefile

```
target  : test1.0 test2.0 test 3.0
          cc-0 target test1.0 to2
test1.0 : test1.0
          cc-c test1.0
test2.0 : test2.0
          cc-c test2.0
test3.0 : test3.0
          cc-c test3.0
```



4) SCCS

การพัฒนาหรือแก้ไขโปรแกรมนั้นจะต้องมีการสำเนาโปรแกรมเดิมไว้ก่อน เพราะหลายครั้งการเปลี่ยนแปลงแก้ไขอาจจะทำให้โปรแกรมแย่ลงกว่าเดิมก็สามารถนำโปรแกรมเดิมมาใช้งานหรือปรับปรุงในทิศทางอื่นอีกได้ ปัญหาที่เกิดขึ้นคือเมื่อมีการพัฒนาหรือแก้ไขโปรแกรมมากครั้งยิ่งขึ้น ทำให้จำนวนไฟล์ที่เป็นสำเนามากขึ้น จนบางครั้งผู้สร้างเองก็เกิดความสับสนว่าไฟล์ไหนแก้ไขเมื่อไหร่ ผู้ใช้บางคนมีหลักการตั้งชื่อก็อาจจะมีการเพิ่มลำดับตัวเลขที่ชื่อไฟล์ว่าเป็นไฟล์แก้ไขลำดับที่เท่าไหร่เมื่อไหร่แต่ถึงกระนั้นก็ตามปัญหาที่เกิดก็คือจะต้องเสียเนื้อที่เดิมข้อมูลในการเก็บทุก ๆ ไฟล์ที่ทำการแก้ไขทั้ง ๆ ที่ส่วนที่แก้ไขหรือเพิ่มเติมอาจจะเป็นส่วนน้อยมากเมื่อเทียบกับไฟล์เดิม

ระบบยูนิกซ์มีโปรแกรม SCCS (Source Code Control System) ช่วยในการจัดการเกี่ยวกับการเปลี่ยนแปลงแก้ไขไฟล์ เพื่อขจัดปัญหาข้างต้น SCCS จะทำการสร้างเลขที่ลำดับของไฟล์ที่แก้ไข โดยเริ่มจากลำดับที่ 1.1, 1.2 ไปเรื่อย ๆ และจะให้ผู้แก้ไขข้อมูลสามารถเขียนคำอธิบายไว้ด้วยรูปแบบการจัดเก็บไฟล์สำเนาของโปรแกรม SCCS นั้นจะเก็บเฉพาะข้อมูลในไฟล์แรกเท่านั้น ส่วนสำเนาลำดับต่อมาจะเก็บเฉพาะส่วนที่มีการแก้ไขเปลี่ยนแปลงเท่านั้น จึงเป็นการประหยัด เนื้อที่การใช้งานในหน่วยเก็บข้อมูล

ลงท้าย

การพัฒนาโปรแกรมบนระบบยูนิกซ์หรือโอเอสใดก็ตาม เป็นสิ่งที่หลีกเลี่ยงไม่ได้เพราะความต้องการของผู้ใช้งานนั้นแตกต่างกันไปตามสาขาของงาน การพัฒนาโปรแกรมบนระบบยูนิกซ์สามารถทำได้ง่ายเนื่องจากระบบได้เตรียมเครื่องมืออำนวยความสะดวกมากมายทั้งยูทิลิตี้ต่าง ๆ ที่มีอยู่ในระบบหรือโปรแกรมที่เอื้ออำนวยความสะดวกในการเขียนและพัฒนาโปรแกรมอย่าง cc make sccs และอื่น ๆ